International Journal of Computer Science Engineering and Information Technology Research (IJCSEITR)
ISSN(P): 2249-6831; ISSN(E): 2249-7943
Vol. 5, Issue 1, Feb 2015, 91-98

HYBRID APPROACH FOR WEB MASHUP SECURITY FRAMEWORK

VINOD SAPKAL[1] & RAVI RANDALE[2]

[1]Department of Computer Engineering, Pillai HOC College of Engineering and Technology, Maharastra, India

[2]Department of Information Technology, Pillai HOC College of Engineering and Technology, Maharastra, India

ABSTRACT

There are several innovative ways for developing software applications on the web. One of the ways is "Web Mashup". It allows users to create new web applications by integrating data and services from other web applications and data sources. Several technologies like Asynchronous JavaScript and XML (Ajax), Rich Site Summary (RSS), Representational State Transfer (REST), and Extensible Markup Language (XML) are used in creating Mashup. The numerous online available data sources and services make Mashup creation fast, easy and also rich in content. Mashup also results in spawning security concerns. A wide array of security issues arise while combining diverse content or services from diverse sources into a new environment. The security issues - User authentication, User/Data confidentiality, Session Fixation, Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), Ajax Vulnerabilities, iframe Security and Brute Force Attack are essential to be addressed. Many research papers aim at providing a security framework for User authentication, User/Data confidentiality and distribution of sensitive information while Session Fixation, Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), Ajax Vulnerabilities, iframe Security and Brute Force Attack are still ignored. The proposed framework system in the project successfully conceptualizes and implements all these attacks.

KEYWORDS: Cryptography, Privacy, Syndication Web 2.0, Mashup Security

INTRODUCTION

A number of new techniques for the creation of web applications have been resulted into the rapid growth of Web 2.0; one of these techniques is meshing up of required content/services from several independent sources for the purpose to create a new content/service [3]. The term "mashup" has been originated from the music field where it means that producing/composing a new song by mixing lyrics and background music from several different source songs [1]. A web mashup can be defined as a situational web application or web page that is created by pulling out, Parsing, and aggregating required relevant content/service from diverse and different publicly available web applications to satisfy user needs and tasks through displaying it to them on their computer screens in a completely different and new way [2]. A web mashup can be created very quickly because it is based on the integration of information already provided by different sources on the web. A simple example can be of a web page extracting map information from one source and location information of a service (such as restaurants or houses for sale or rent) from another site and display the map with services placed on it as shown in Figure 1 [7]. Maps mashup that lets explore locations with helpful information including weather, photos, facts, events, news, and much more from the popular sources like WeatherBug, Flickr, Wikipedia, Eventful, and Google Based on the sources and nature of information to be Integrated, a web mashup can be classified as either Consumer mashup or Data mashup or Business mashup[1]:

- A data mashup combines similar types of data /information from diverse sources into a unique representation and create a new web service.
- A business mashup allows organizations to define web applications that combine their own internal resources such as application and data with some external enhanced web services. Business mashupsare also called Enterprise mashups. They put the information into a single presentation, provide a collaborative environment for both businesses and developers and are very rich as well as secure applications.
- A consumer mashup combines different data types and uses public web sites that made their content available through well-defined APIs and feeds, thus resuming less programming expertise.

Architecturally (on the basis of locations of content integrator and content consumer), a web mashup can be either Web - Based or Server - Based [5]

- News mashups use RSS or A TOM syndication and disseminate news feeds to the users. For example: DiggDot.Us9
- Searching and Shopping mashups search different content providers for a product, compares their offered prices and comes up with an optimum result to the user. eBay aznd Amazon have provided their APIs for accessing their content programmatically
- Video and Photo mashups combine other information with videos or photos using the metadata associated with videos or photos. For example, Google Maps in Flicker (GMiF)
- Mapping mashups uses maps and location data to be graphically displayed, and is the most popular type of mashup as shown in Figure 1.

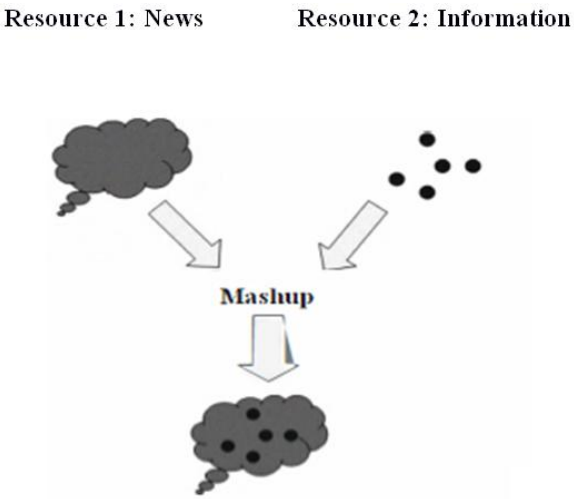

Figure 1: Architecture of Web Mashup [1]

WEB MASHUP SECURITY

The security model Same-Origin Policy implemented by browsers also creates problems for web mashups security. SOP uses origin to classify documents: documents belonging to the same origin can openly access each other's

contents whereas documents from different origins are not allowed; origins are recognized by hostname/lntemet domain, protocol along with port [4]:Web mashup is in its infancy, lacking robust standards, protocols and toolkits to facilitate mature software development process. Besides the several technical, component and social issues [1], web mashups being primarily concerned with integrating content/services from different sources into a new web service, therefore introduce new security issues. These include [5]: who made available the information to be combined, what are the intentions behind the information availability (e.g. commercial reasons etc.,), and what is the trustworthiness of the content available? Security issues can arise either from the security problems or loop-holes of the technology being used or the nature of the mashup to be performed [5].

- Scripts' access to cookies and browser plug-ins is restricted by the Same-Origin Policy
- One origin script should not use XMLHttpRequest to communicate with a site belonging to another origin..
- One origin script should not be able to access the JavaScript environment of a frame belonging to another origin.
- One origin script should not be able to read or write content to/from another document or frame belonging to another origin.

Malicious code from an intruder site bypasses web browser by originating request through an authenticated browser page to a trusted site (server) and the responses are unknowingly transmitted back to the intruder site [1]. Techniques used to create web mashups completely ignores the Same-Origin Policy and a web application violating the SOP would be a good working example of web mashup as an integrator would combine information from different content providers belonging to different trust domains. Hence a new security framework is required to be built where issues like user privacy,data integrity, data confidentiality, and user authentication are properly addressed [4].iframe tag, which creates a single, atomic DOM node in the parent document [4], was introduced to solve the problems associated with SOP [5]. However, there still exists several ways of attacking browsers, the most wellknown are Cross-Site-Scripting (XSS) and Cross-SiteRequest- Forgery (CSRF) [3]. XSS is usually the result of session fixation which occurs when a new user's session is authenticated at the server without invalidating the previous/existing session. An intruder can violate this situation by capturing session credentials and redirecting and storing sensitive data CSRF takes place

- **Data Confidentiality**

The communicating entities may wish to guarantee that the content being exchanged should be readable to the intended parties only, any other third party on the way should not be able to disclose content of the packets being exchanged.

- **User Authentication**

The communicating entities may wish to guarantee and verify the identities of each other to build trust measures.

- **Data Confidentiality**

The communicating entities may wish to guarantee that the content being exchanged should be readable to the intended parties only, any other third party on the way.

IMPLEMENTED SYSTEM

Many research papers aim at providing a security framework for User authentication, User/Data Confidentiality and distribution of sensitive information while Session Fixation, Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), Ajax Vulnerabilities, iframe Security and Brute Force Attack are still ignored. The implemented system is project successfully conceptualizes and implements all these attacks.

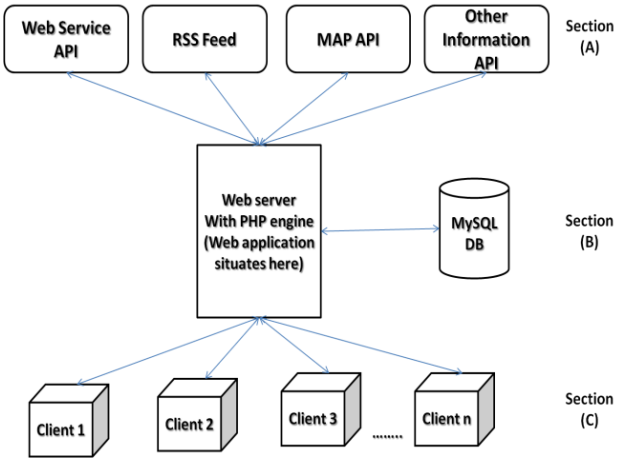

Figure 2: Implemented System Architecture Model

- **User Authentication**

There is possibility that an invalid user can access the web Mashup application. So to avoid that, in the proposed Mashup system only accept valid username and password. Therefore only valid user can access Mashup application and invalid user will be restricted.

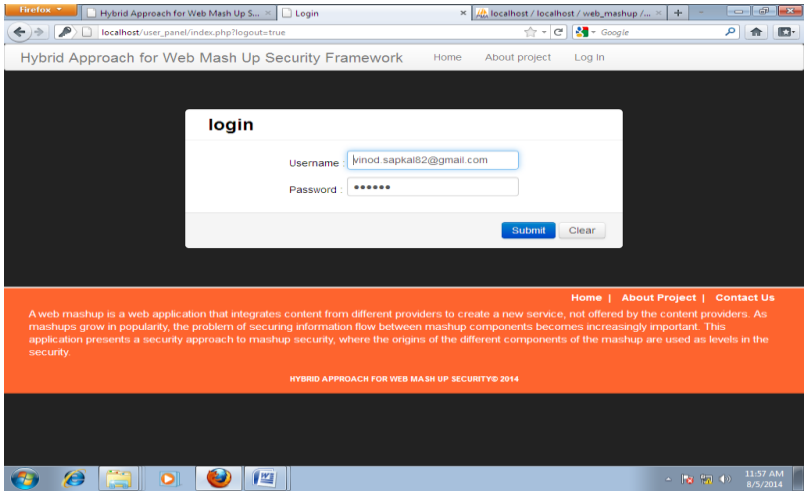

Figure 3

- **User/Data Confidentiality**

Authenticated User may put important or confidential information. Intruders can capture or hack such information, hence to avoid such action sensitive data is converted into encrypted code. And only authenticated user can access confidential data.

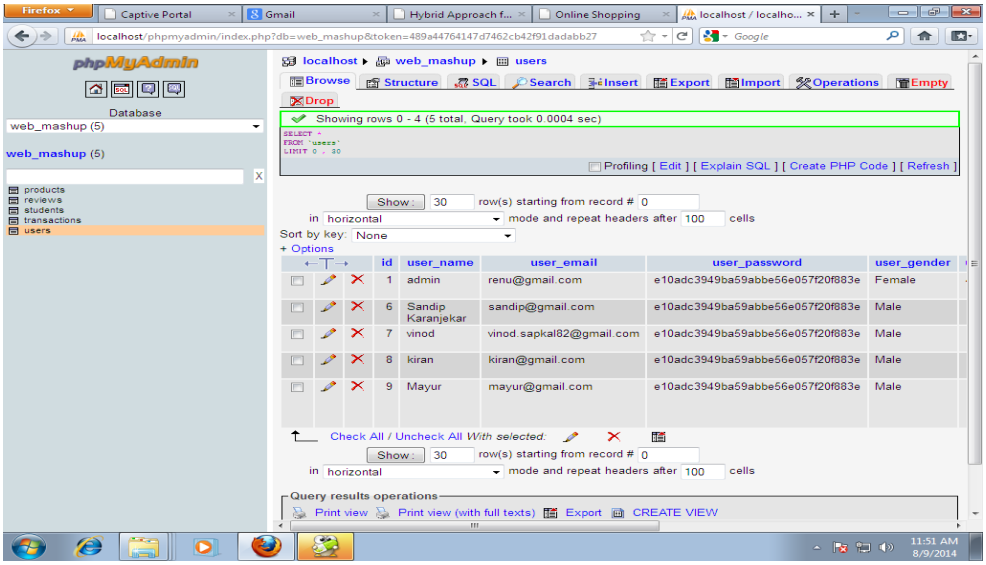

Figure 4

- **Preventing Session Fixation**

When authenticated user does not validate existing sessions then it leads to *Session Fixation.* Session Fixation allows intruders to intercept authenticated sessions or to create new sessions. To avoid Session Fixation, each session will have unique session identifier.

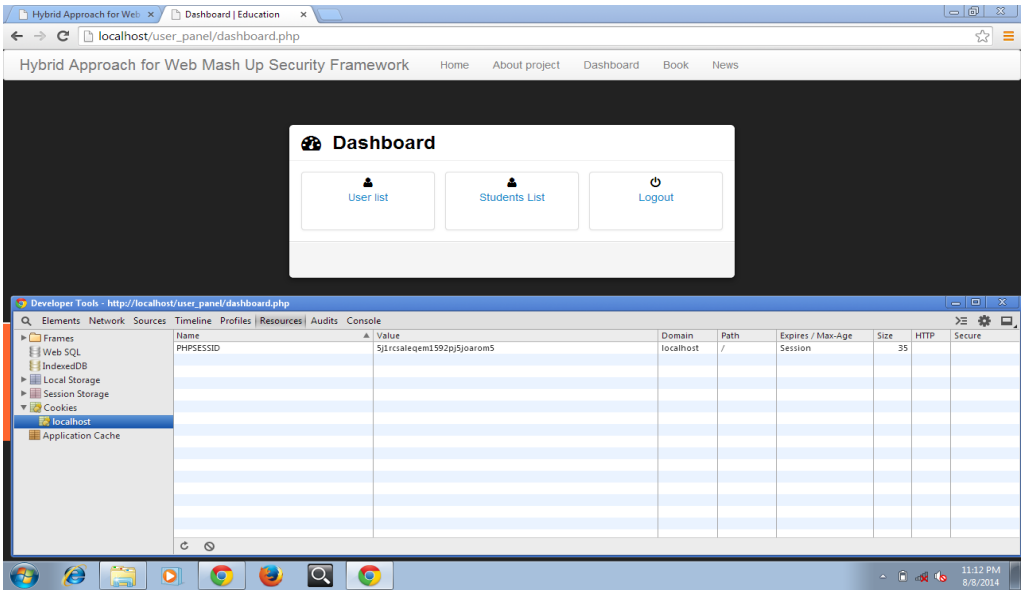

Figure 5

- **Preventing Cross Site Scripting (XSS) Attack**

One technique used in many Mashup is to embed JavaScript snippets that are dynamically downloaded and interpreted on demand. On-demand scripts can include malicious code aimed at exploiting security vulnerabilities such as XSS. One can prevent this vulnerability by ensuring that on-demand scripts are validated and that content generated from the scripts is encoded properly to prevent execution of malicious code.

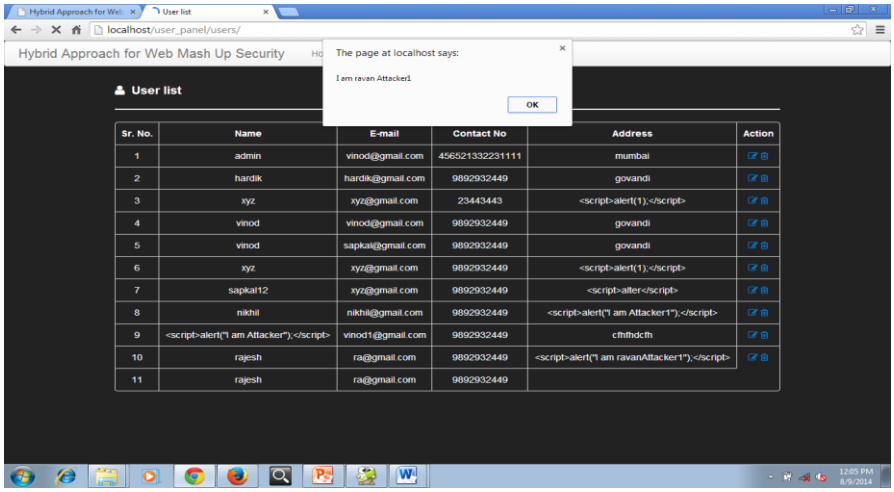

Figure 6

- **Preventing CSRF attacks (Man-In-Middle Attack)**

In CSRF attack requests originate from an intruder site and are transmitted through an authenticated browser page to the server. Responses are then transmitted unknowingly back to the intruder site. A technique for preventing CSRF attacks that Mashup servers often use is to require each HTTP request to include a request-specific token to be transmitted to the server with each POST and GET request.

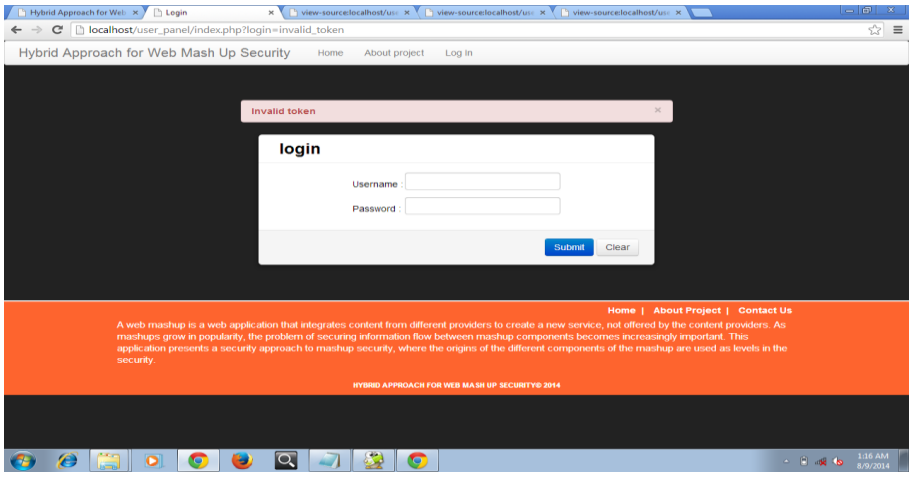

Figure 7

- **Iframe Security**

Mashup widgets are often embodied as embedded inline frames (iframes). *iframes* are HTML elements regarded as separate entities by a browser within a page. Content placed inside an iframe cannot manipulate any part of a browser's Document Object Model [5] (DOM) outside of the containing iframe. The proposed system blocks such iframes.

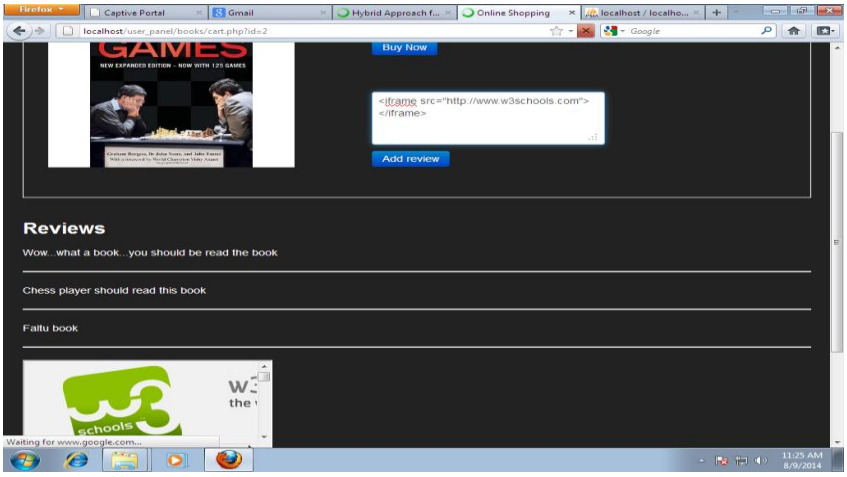

Figure 8

- **Brute Force Attack**

In Brute Force Attack, attacker tries various combinations of Username and Password to get access of website. But in the proposed system, if an attacker tries to implement Brute Force Attack then that username is blocked.

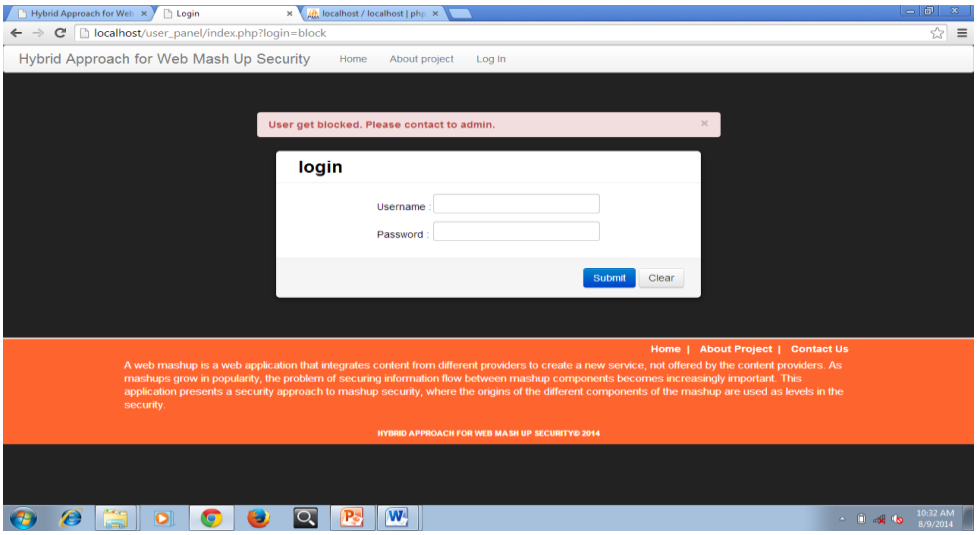

Figure 9

CONLUSIONS AND FUTUREWORK

The existing system is addressing issues like User authentication, User/Data confidentiality and distribution of sensitive information while Session Fixation, Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), Ajax Vulnerabilities, iframe Security and Brute Force Attack are still ignored. Considerations of these issues make a top security priority while developing a Mashup Application. So in order to overcome these ignored issues new proposed system has implemented. This proposed system not only provide security to issues like User authentication, User/Data confidentiality, Session Fixation, Cross Site Scripting (XSS), Cross Site Request Forgery (CSRF), Ajax Vulnerabilities, iframe Security and Brute Force Attack but also improve the performance of Web Mashup Security. There are certain other parameters such as access control and non-repudiation which are also essential for a system's security. These aspects are not covered in the current proposed solution and are left for the future extension

ACKNOWLEDGEMENTS

Authors are greatful and thankful to Dr. Chelpa Lingam, Prof. Sandeep Raskar and Prof. Rohini Bhosale for assistance to carry out this research work**.**

REFERENCES

1. Shaukat Ali, Shah Khusro and Azhar Rauf, "A Cryptography-Based Approach to Web Mashup Security", IEEE International Conference, Vol. 4, Pp.125-130, 2013.
2. Web Mashup[Available on] http://en.wikipedia.org/wilki/Mashup_%28web application_hybrid%29.
3. Saman Zarandioon, Danfeng (Daphne) and Yao Vinod Ganapathy, "OMOS: A Framework for Secure Communication in Mashup Applications", Annual Computer Security Applications Conference, Vol. 2, Pp. 155-160, 2008.
4. Chen Yanchun, Wang Xing peng, "A Security Risk Evaluation Model for Mashup Application**",** International Conference on Information Management, Innovation Management and Industrial Engineering, Vol. 3, Pp. 1233–1237, 2009.
5. Gerald Bader, Amin Anjom shoa and Min Tjoa "Privacy Aspects of Mashup Architecture"**,** IEEE International Conference on Social Computing / IEEE International Conference on Privacy, Security, Risk and Trust, Vol. 43, Pp. 150–160, 2010.